Digging in to the details of SSL/TLS

Stephan Schulz – Specialist SE Security

Frank Thias – Principal SE

Agenda

- What is SSL/TLS
  - Where is SSL (osi)
  - Challanges
- SSL/TLS – basic facts
  - Example – full handshake
  - TPS and Ciphersuites
- Drivers – aside from Security
  - Industry trends
  - SSL-everywhere
  - Perfect Forward Secrecy - PFS
- SSL/TLS – digging deeper
  - ECC and cipher diversity/support
  - HSTS, Public Key Pinning
- SSL/TLS debugging
  - Negotiations
  - Error codes
- Key Management
- Use Cases
  - Reverse- / Forward proxies
  - SSL/TLS Intercept
- Summary

What is SSL/TLS

What is SSL?

The history of SSL and TLS

SSL1 and SSL2
Created by Netscape and contained significant flaws

SSL3
Created by Netscape to address SSL2 flaws

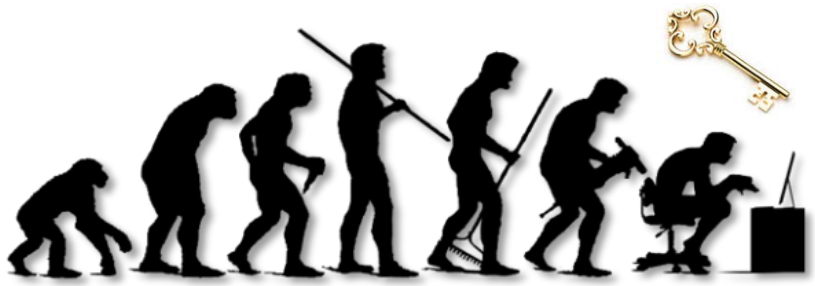
TLS 1.0
Standardized SSL3 with almost no changes
RFC2246

TLS 1.1
Security fixes and TLS extensions
RFC4346

TLS 1.2
Added support for authenticated encryption (AES-GCM, CCM modes) and removed hard-coded primitives
RFC5246

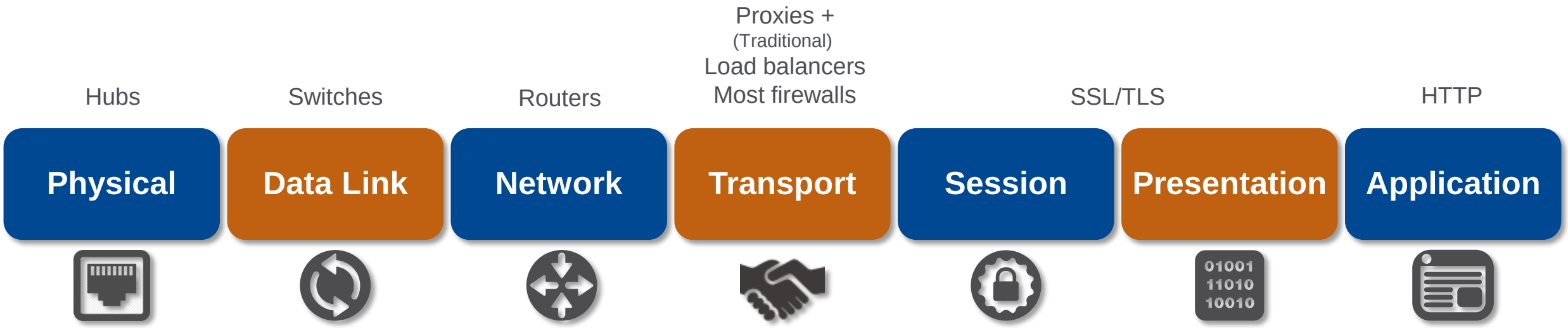
1994 | 1995 | 1999 | … | 2006 | 2008

Crap hits the fan
First set of public SSL exploits



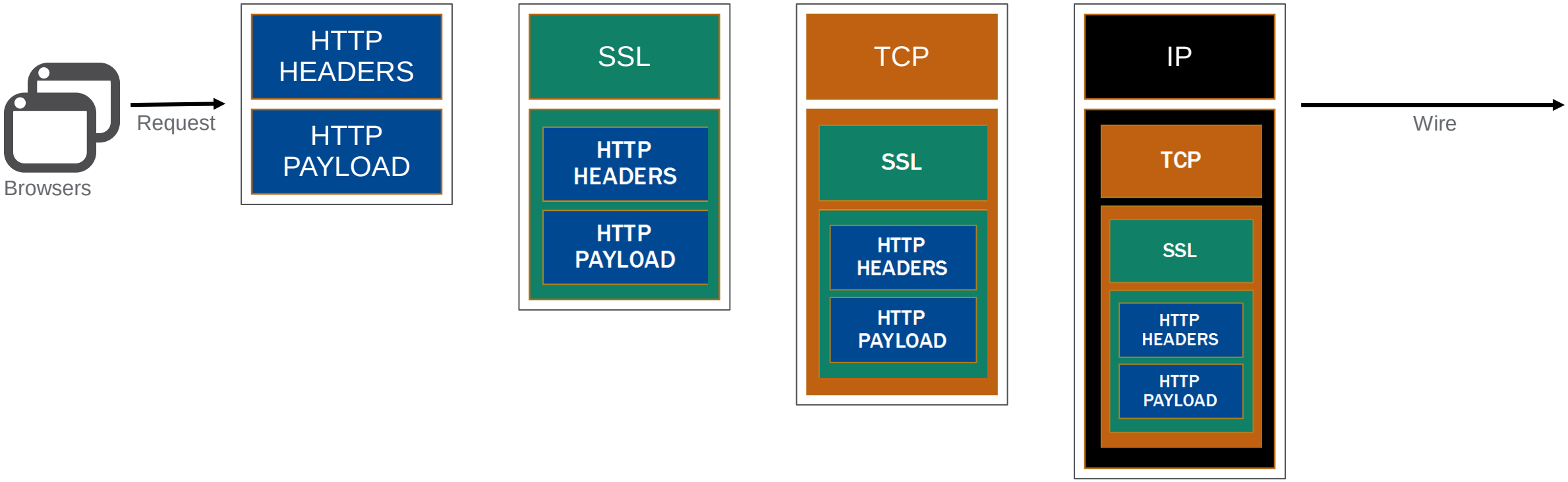
Where is SSL?

OSI Model



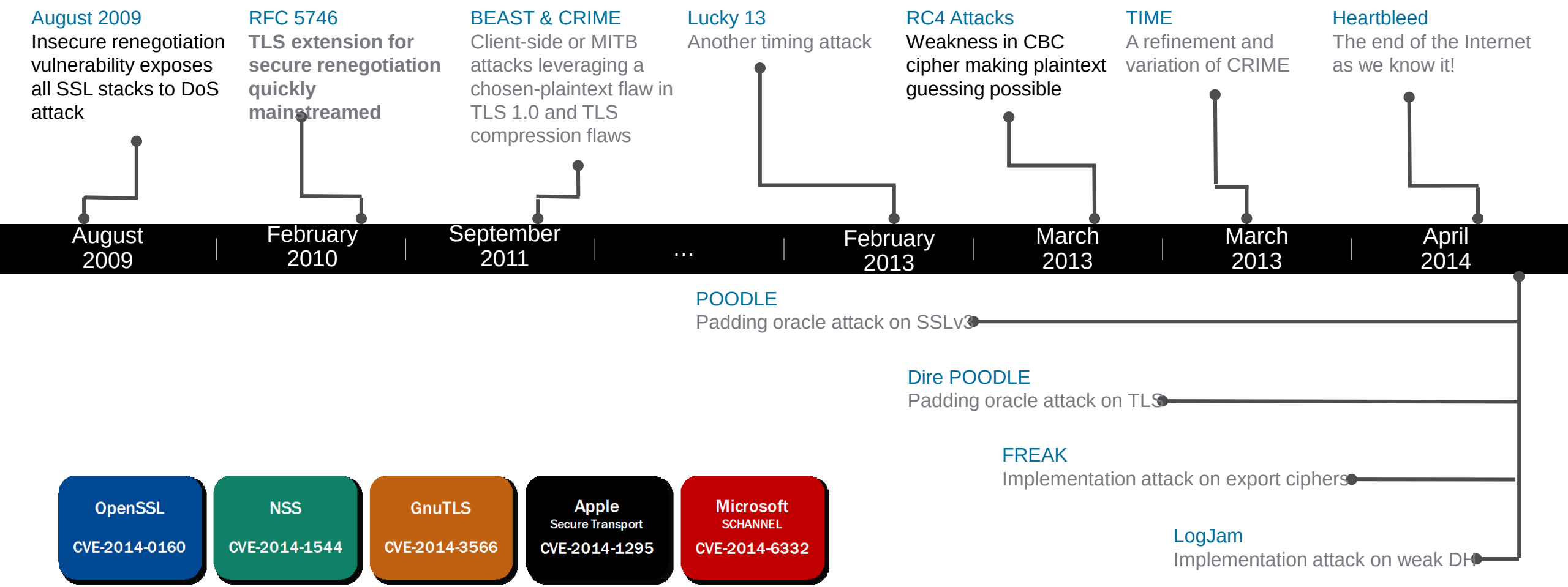
Where is SSL?

Encapsulation



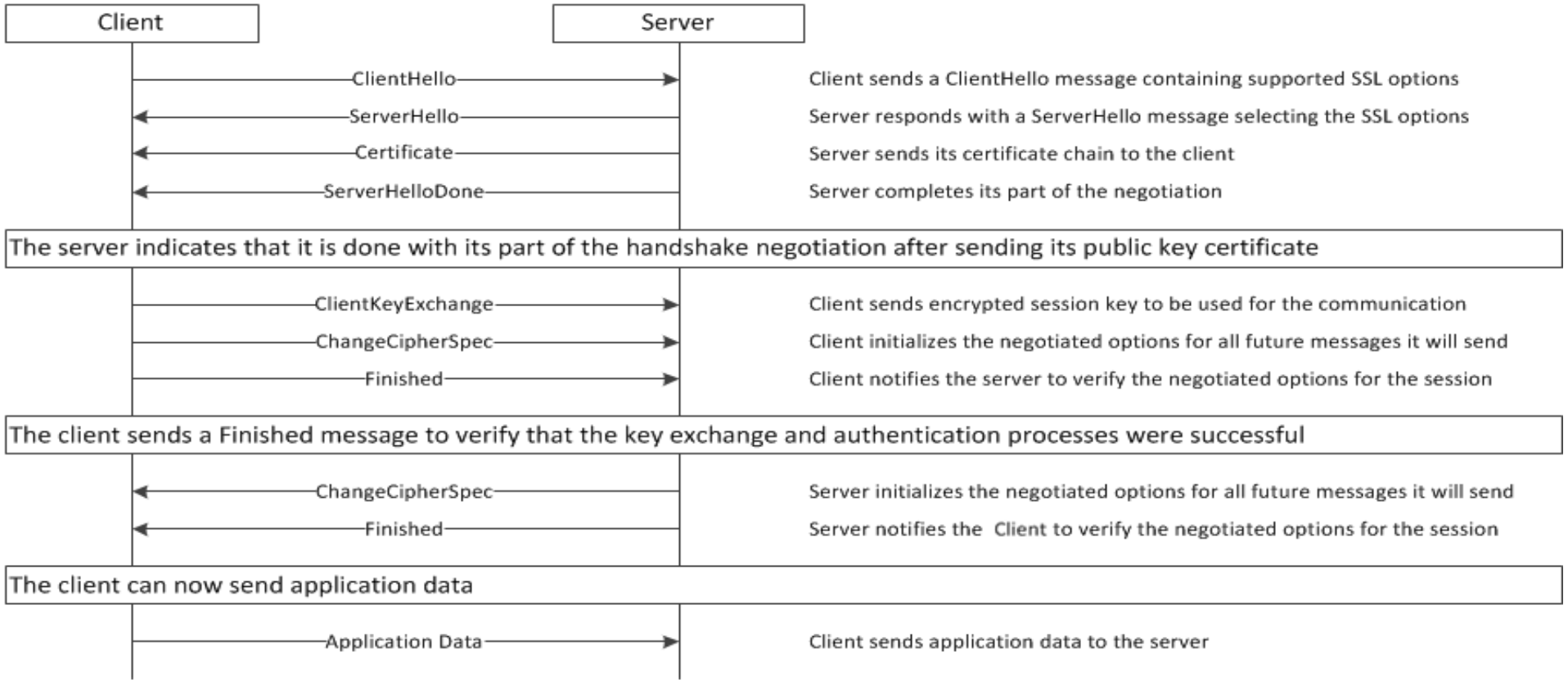
SSL isn't perfect

SSL vulnerabilities exposed



SSL/TLS – basic facts

SSL Handshake



- RSA, EC, DH including presentation of Cert
- One handshake counts as one TPS
- Bulk Encryption - measured as throughput, e.g. 1 Gb/s

SSL/TLS – TPS

Transactions per Second

- TPS - key exchange and handshaking capability of a device
- handshake operations for every new SSL session
- computationally-intensive
- specialized hardware available

- Different implementations possible
- Specialized HW only used for key exchange and handshake (higher number)
- Specialized HW used for key exchange, handshake and ongoing encryption (lower number but higher Bulk Encryption Throughput)

Driver for SSL/TLS – aside from Security

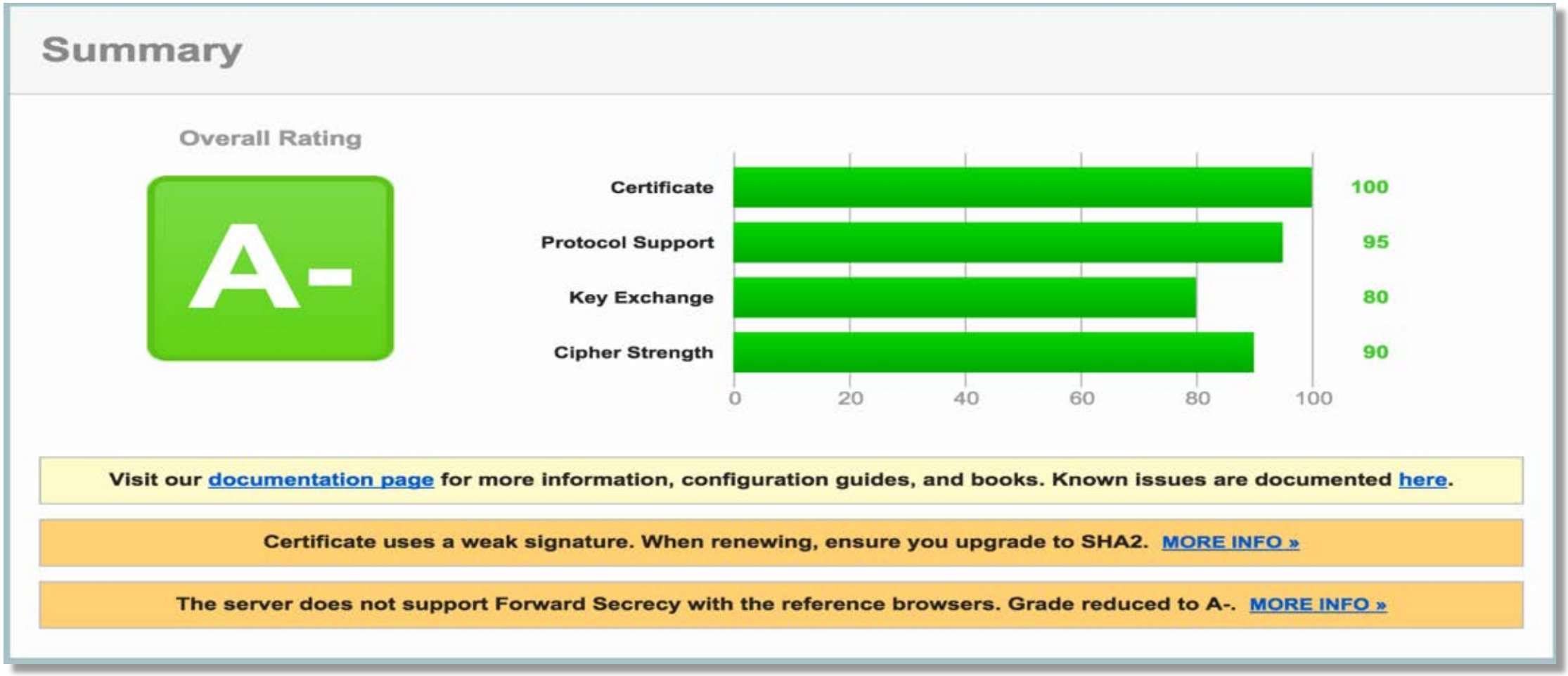
Industry trends

External pressures are mounting

- Google now using HTTPS as a ranking signal. http://googlewebmastercentral.blogspot.com/2014/08/https-as-ranking-signal.html
- PCI DSS 3.1 now requires TLS1.1+ for all new deployments. https://www.pcisecuritystandards.org/documents/PCI_DSS_v3-1.pdf
- Firefox is deprecating HTTP! https://blog.mozilla.org/security/2015/04/30/deprecating-non-secure-http/
- And HTTP/2 is only supported over TLS.

Industry trends

Achieving the coveted SSLLabs A+ rating



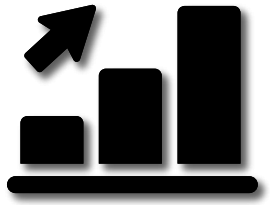
SSL Everywhere is

a fundamental paradigm shift in the natural state of IT services, by which nothing is trusted and everything is encrypted by default.

Encryption creates blind spots

Malware

uses encrypted channels to evade detection



Visibility

is reduced due to the growth of SSL usage

Performance

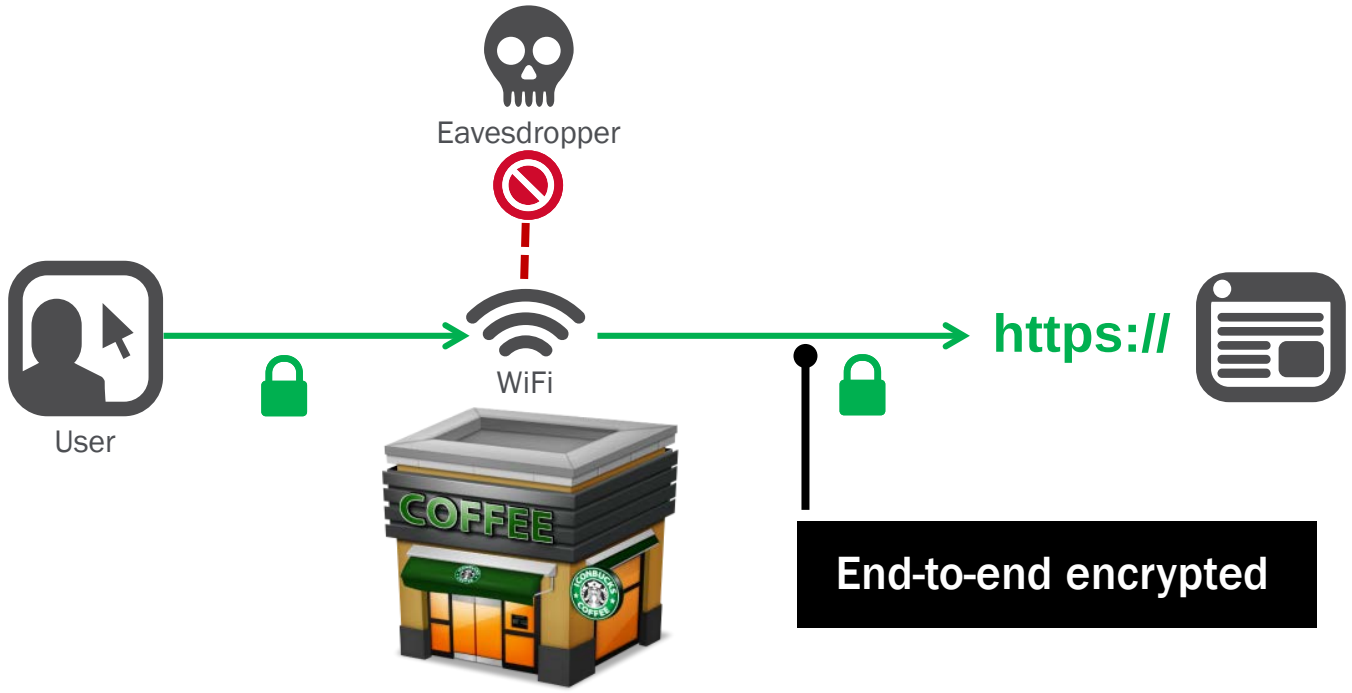
for decryption is a significant undertaking

Enabling SSL on a firewall or an IPS can reduce the overall performance of the appliance, often by more than 80%

Advancing to SSL Everywhere

F5 Solution

- Encrypt by default
- Performance and scale
- Best-in-class cryptography
- Integrated partner solutions
- Protect in data center, cloud, and hybrid environments



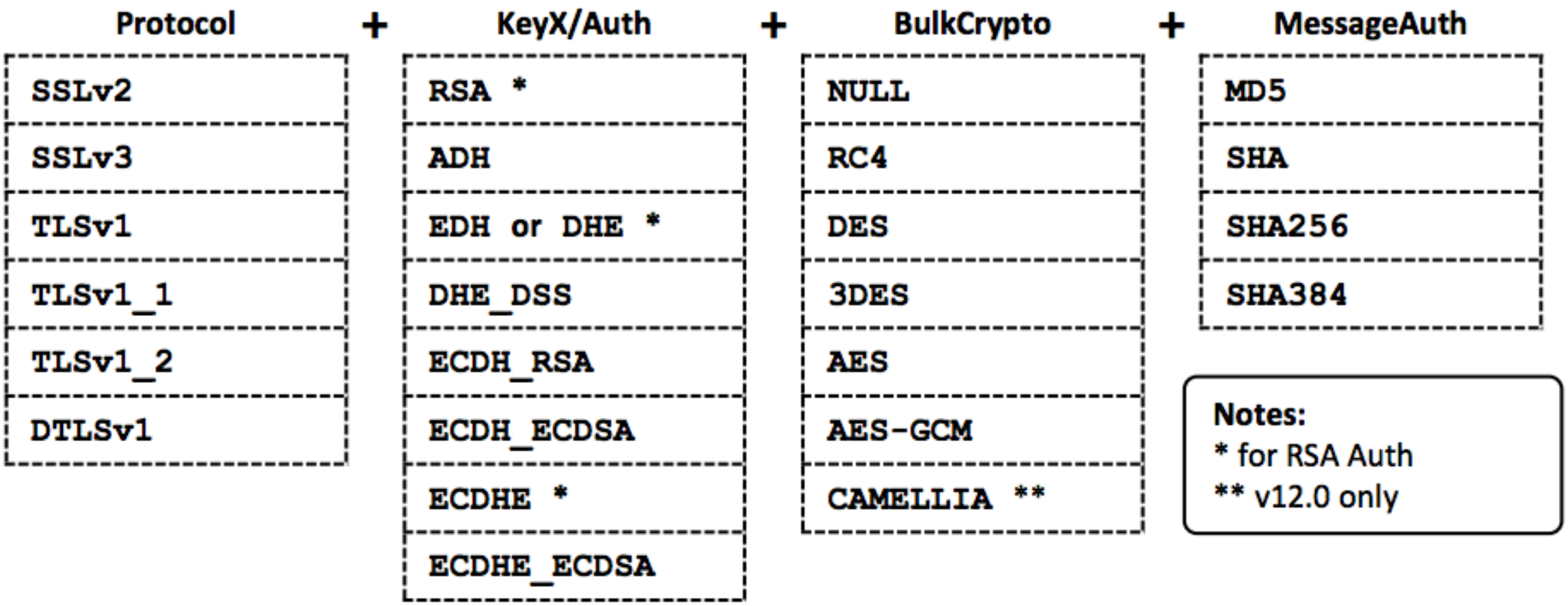
Important…

passive SSL inspection (IPS/WAF/NGFW/...) require RSA-style session establishment because they work by doing a man-in-the-middle using the server's private key. (This includes passive span-mode and bridge-mode devices.)

With PFS only SSL termination /re-encryption will provide Visibility

SSL/TLS – digging deeper

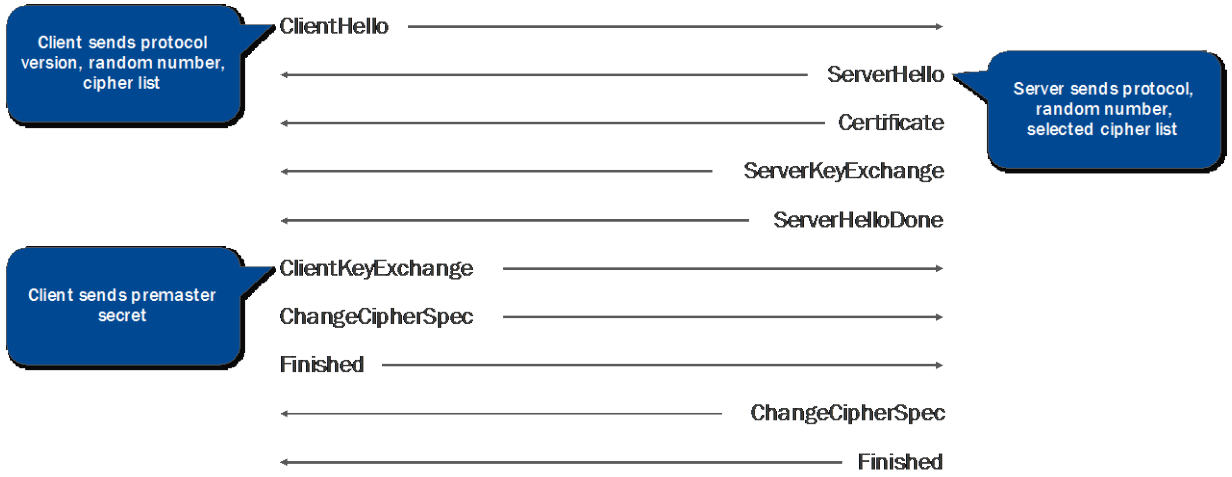
SSL/TLS Cipher Suites



https://devcentral.f5.com/questions/tmos-ssl-tls-cipher-cheat-sheet

Perfect forward secrecy (PFS) – Why?

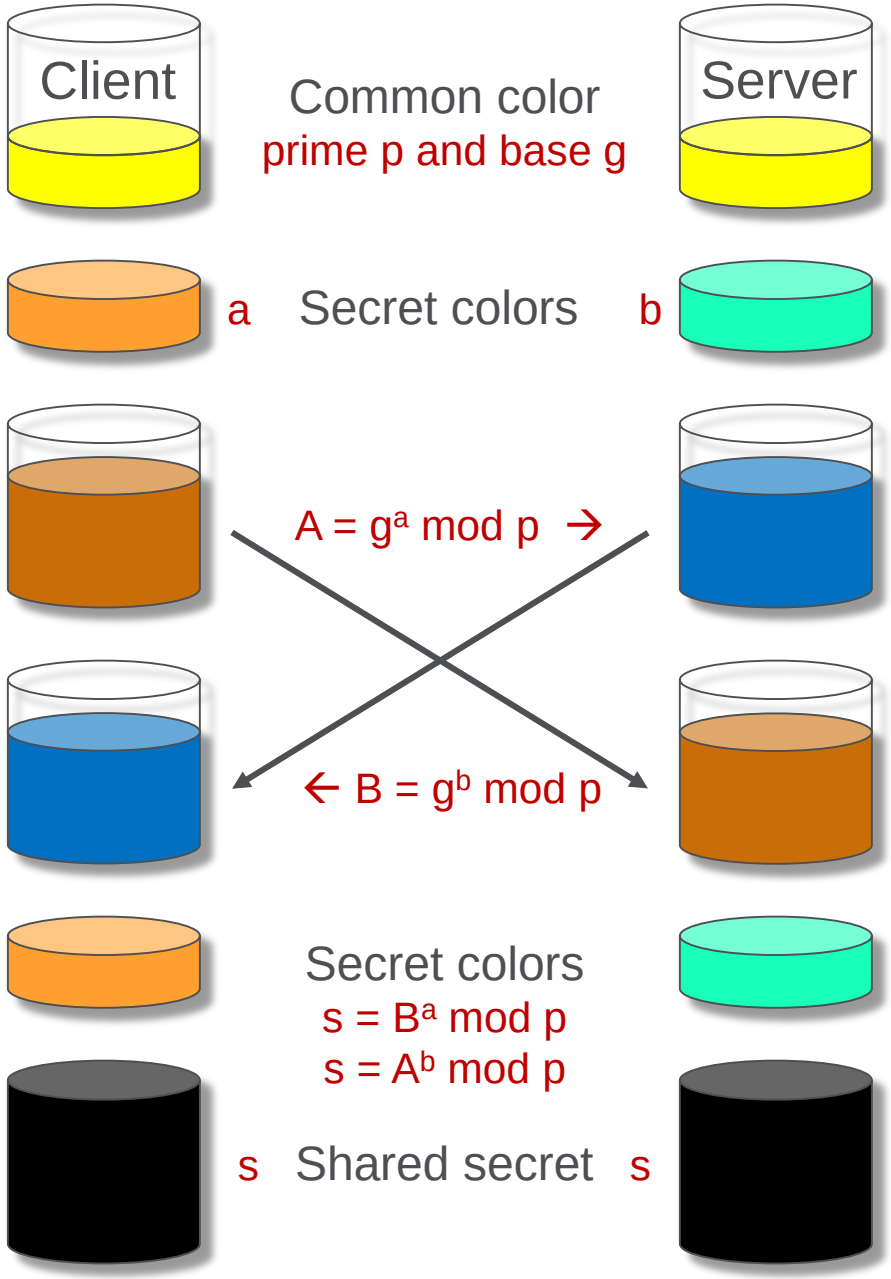
- RSA key exchange
- Client sends a random value
- Server sends a random value



- Client sends a pre-master secret encrypted with server's public key
- Server's key is used for authentication **AND** encryption
- **If the server's key is ever compromised, all previous messages can be decrypted**

Perfect forward secrecy (PFS)

- Diffie-Hellman key exchange
  - Client and server agree on a prime and base
  - Client/Server generates a secret and performs a modulo function with that value, then sends this value to each other
  - Client and server separately compute the shared secret
  - The server's key is only used for authentication
  - If the shared secret is changed for every session, it's considered "ephemeral" (DHE). Compromise of the server's private key will not allow decryption of previous messages



Elliptic Curve Cryptography (ECC)

- Next generation of public key cryptography
- Uses the mathematical properties of elliptic curves to provide better security with smaller key sizes
  - ECC TLS ciphers are defined in RFC 4492
- Many defined but only two widely supported
  - secp256r1/NIST P-256/prime256v1
  - secp384r1/NIST P-384

```
enum {
        sect163k1 (1), sect163r1 (2), sect163r2 (3),
        sect193r1 (4), sect193r2 (5), sect233k1 (6),
        sect233r1 (7), sect239k1 (8), sect283k1 (9),
        sect283r1 (10), sect409k1 (11), sect409r1 (12),
        sect571k1 (13), sect571r1 (14), secp160k1 (15),
        secp160r1 (16), secp160r2 (17), secp192k1 (18),
        secp192r1 (19), secp224k1 (20), secp224r1 (21),
        secp256k1 (22), secp256r1 (23), secp384r1 (24),
        secp521r1 (25),
        reserved (0xFE00..0xFEFF),
        arbitrary_explicit_prime_curves(0xFF01),
        arbitrary_explicit_char2_curves(0xFF02),
        (0xFFFF)
    } NamedCurve;
```

	Minimum size (bits) of Public Keys			Key Size Ratio	Protection from
Security (bits)	DSA	RSA	ECC	ECC to RSA/DSA	Attack
80	1024	1024	160-223	1:6	Until 2010
112	2048	2048	224-255	1:9	Until 2030
128	3072	3072	256-383	1:12	Beyond 2031
192	7680	7680	384-511	1:20	
256	15360	15360	512+	1:30	

Table 1: National Institute of Standards and Technology Guidelines for Public-Key Sizes

SSL cipher diversity

Cryptographic feature set, updates and PFS

Onboard HSM	Always
TLS 1.2	10.2.3
SHA256	10.2.3
RFC5726 Secure Renegotiation	10.2.3
Network HSM	11.2.1
DSA	11.4.0
ECDH and ECDHE	11.4.0
Perfect Forward Secrecy	11.4.0
ECDSA	11.5.0
AES-GCM	11.5.0
ECC	11.5.0
TLS_FALLBACK_SCSV	11.4.1HF6 11.5.1HF6 11.5.2 11.6.0

Removed SSLv3 from DEFAULT	11.5.0
Added DHE to DEFAULT	11.6.0
Removed RC4 from DEFAULT	11.6.0
Re-added DHE-RSA/DHE-DSS with DTLS	11.6.0
Added “!” vs “–” for SSLv3 in DEFAULT	11.6.0
Moved ADH from COMPAT to NATIVE	11.6.0

Perfect Forward Secrecy

- **Offered by the Diffie-Hellman key exchange (DHE/ECDHE)**
- **Uses a different key for each connection**
- **Prevents decryption of a recorded SSL session if a private key is compromised**
- **Requires TLS and an appropriate cipher suite**

V12.0.0 - Camellia Cipher support

- Cipher which is seeing increased use in EU and Japan
- Part of the Native cipher suite
  - DHE-RSA-CAMELLIA256-SHA
  - DHE-RSA-CAMELLIA128-SHA
  - CAMELLIA256-SHA
  - CAMELLIA128-SHA
  - DHE-DSS-CAMELLIA256-SHA
  - DHE-DSS-CAMELLIA128-SHA

SSL cipher diversity

Cipher management

NATIVE

Supported by TMM codec and TLS stack

- Contains cipher suites optimized for the BIG-IP
- Faster than the COMPAT stack
- However not all ciphers are hardware accelerated

DEFAULT

Subset of NATIVE ciphers – restricted by rules

!LOW:!SSLv3:!MD5:!RC4-SHA:!EXPORT:DHE+AES-GCM:DHE+AES:DHE+3DES:AES-GCM+RSA:RSA+AES:RSA+3DES:ECDHE+AES-GCM:ECDHE+AES:ECDHE-RSA-DES-CBC3-SHA

COMPAT

Supported by TMM's codec and OpenSSL stack

- Supports all of the NATIVE ciphers
- And supports ciphers from the OpenSSL suite
- Slower than the NATIVE stack and requires more memory
- Makes minimal use of hardware acceleration

ALL

Effectively the same as NATIVE

HTTP Strict Transport Security - HSTS

- Defined in RFC6797
  - Inserts a header – instructs the browser to use HTTPS only to a domain (and. optionally, any subdomains) for a period of time
  - To mitigate MiTM attacks - manipulation of redirects or step-down strategies (sslstrip / firesheep)
  - Supported by many modern browsers (Firefox 4, Safari (Mac OS X 10.9), Opera 12, Chrome 40.0.211.0, IE12 (Windows 10)
  - an iRule prior to 12.0
  - a configurable HTTP profile object in v12.0

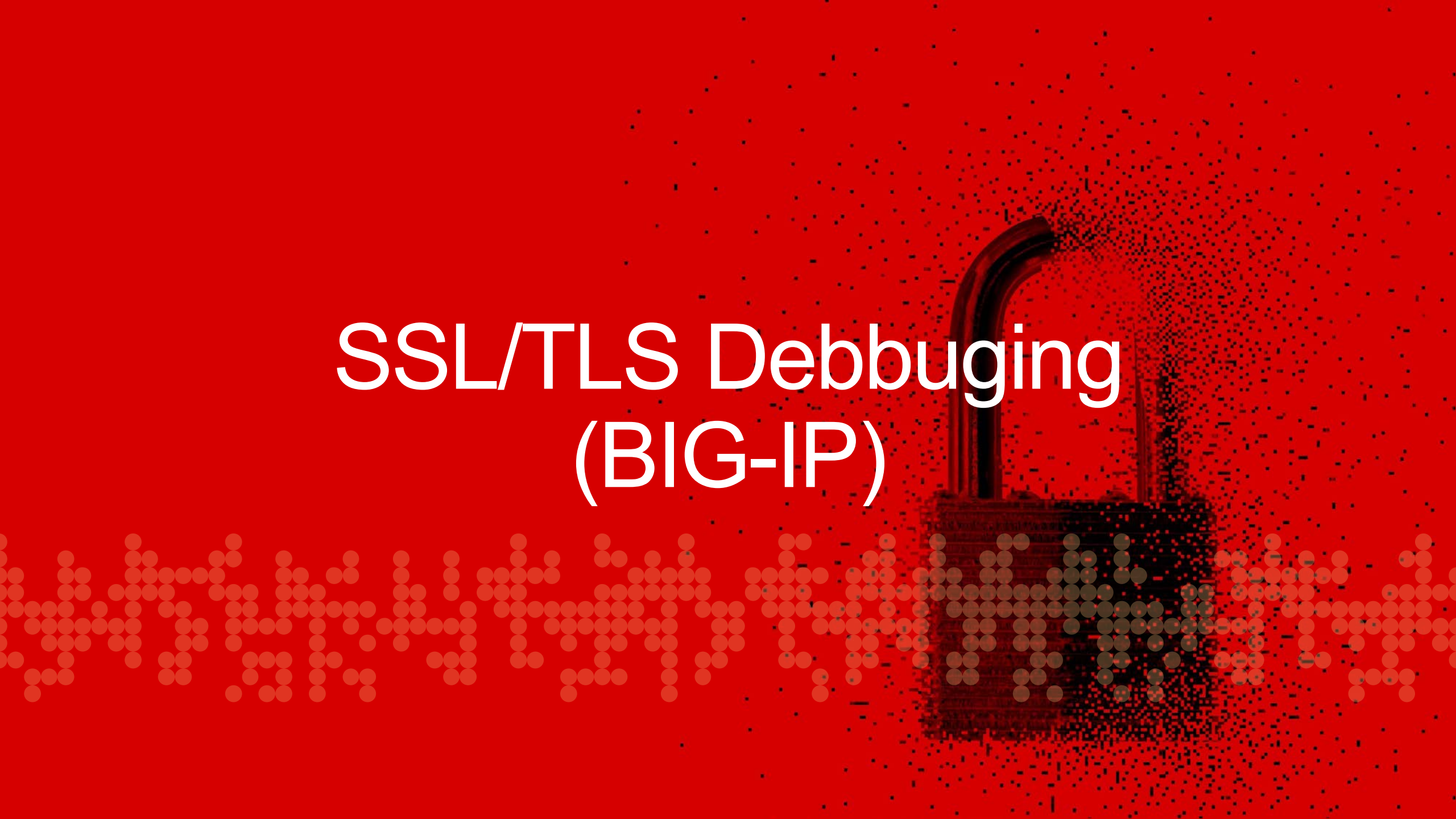
HTTP Public key pinning - HPKP

- Detects changes of a public key of a certificate for a specific host
  - Browser looks up any stored pins
  - Terminates the connection if pin validation fails

- Requires a backup strategy
  - Browser will accept only the specific identities we provide in the header
  - It's recommended to create some backup CSRs and include their fingerprints in the header

- An iRule can be used to add the header
  https://devcentral.f5.com/questions/configure-public-key-pinning-hpkp-header-in-ltm-116
  https://devcentral.f5.com/questions/insert-multiple-public-key-pins-with-irule

SSL/TLS Debbuging (BIG-IP)



Successful negotiation

```
1 1 0.0003 (0.0003) C>SV3.3(79) Handshake
ClientHello
Version 3.3
cipher suites
TLS_RSA_WITH_RC4_128_SHA
TLS_RSA_WITH_AES_128_CBC_SHA
TLS_RSA_WITH_AES_256_CBC_SHA
TLS_RSA_WITH_AES_128_CBC_SHA256
TLS_RSA_WITH_AES_256_CBC_SHA256

1 2 0.0008 (0.0005) S>CV3.1(74) Handshake
ServerHello
Version 3.1
cipherSuite TLS_RSA_WITH_RC4_128_SHA
```

- The client offered protocol TLSv1.2 (version 3.3) and the server downgraded the protocol to TLSv1.0 (version 3.1)
- The server also chose the preferred cipher from the client's list

Unsuccessful negotiations...

```
1 1 0.0012 (0.0012) C>SV3.0(47) Handshake
ClientHello
Version 3.0
cipher suites
SSL_RSA_WITH_AES_256_CBC_SHA

1 2 0.0013 (0.0000) S>CV0.0(2) Alert
level fatal
value handshake_failure
```

- Protocol negotiation unsuccessfully
  - server does not support protocol version below TLS1 (version 3.1) and the client does not support protocol versions above SSLv3 (version 3.0):

```
1 1 0.0012 (0.0012) C>SV3.1(58) Handshake
ClientHello
Version 3.2
cipher suites
TLS_DH_anon_WITH_RC4_128_MD5

1 2 0.0013 (0.0000) S>CV3.2(2) Alert
level fatal
value handshake_failure
```

- Cipher negotiation unsuccessfully
  - the server does not support any of the client's ciphers.

SSL error codes for /var/log/ltm

The following table provides common SSL error codes and possible explanations for the error:

Error code	Message	Description
10	unexpected_message	The peer received an inappropriate SSL message.
20	bad_record_mac(20)	The peer received a record with an incorrect MAC.
22	record_overflow	The peer received a record that exceeds the maximum number of bytes.
30	decompression_failure	The decompression function received improper input.
40	handshake_failure	The sender was unable to negotiate an acceptable set of security parameters given the options available.
42	bad_certificate	The certificate was corrupt or contained signatures that could not be correctly verified. This alert can occur if the client certificate was signed by a different CA than the one specified in the SSL profile.
43	unsupported_certificate	The certificate type was unsupported.
44	certificate_revoked	The certificate was revoked.
45	certificate_expired	The certificate was expired.
46	certificate_unknown	An unspecified issue occurred while processing the certificate.
47	illegal_parameter	A field in the handshake was out of range or inconsistent with other fields.
48	unknown_ca	A valid certificate chain or partial chain was received, but the certificate was not accepted because the CA certificate could not be located or matched with a known, trusted CA.
49	access_denied	A valid certificate was received, but when access control was applied, the sender decided not to proceed with negotiation.
50	decode_error	A message could not be decoded because some field was out of the specified range, or the length of the message was incorrect.
51	decrypt_error	A handshake cryptographic operation failed, including being unable to correctly verify a signature or validate a **Finished** message.
70	protocol_version	The protocol version the client attempted to negotiate was recognized but not supported.

Debugging

- Enable ssl debugging
  - **`tmsh modify /sys db log.ssl.level value Debug`**

    and review /var/log/ltm

  - ***Important****: After you test SSL connections for the virtual server using a web browser or OpenSSL client, you should disable SSL debug logging by typing the following command:*
    `modify /sys db log.ssl.level value Warning`

- Test ssl-connection with openssl and s_client, e.g.:
  - **`openssl s_client -connect 10.255.0.202:443`**
  - After this issue a **`GET / HTTP/1.0`** if the VS is a webapp

tcpdump and ssldump for debugging

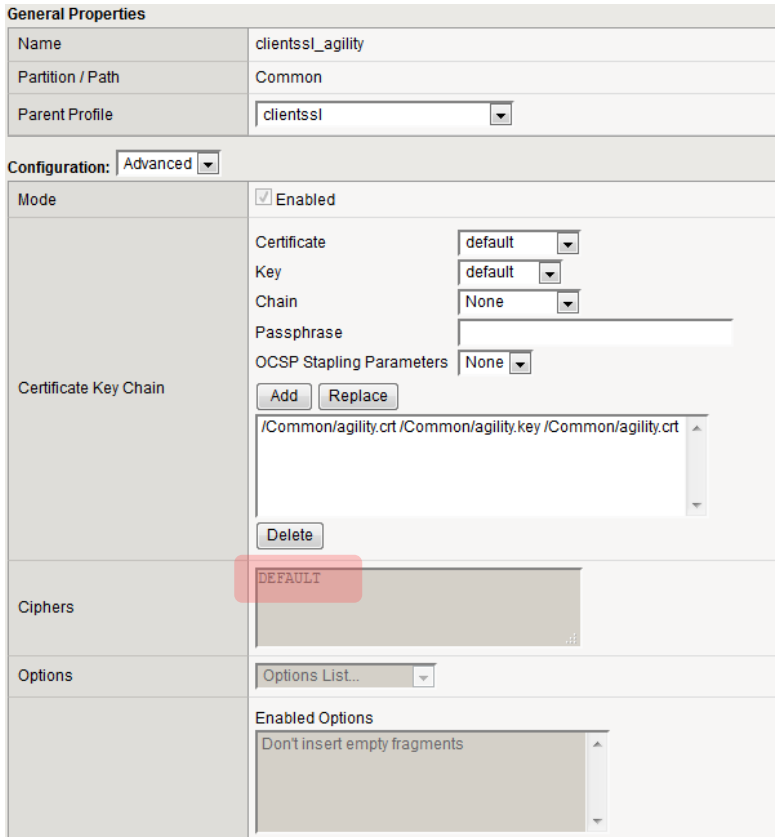
- tcpdump -i 0.0 host 10.255.0.202 -n -s0 -w /tmp/ssl.cap
- ssldump -r /tmp/ssl.cap
  - -k switch could be used to include the private key and display decrypted data
  - Note: This works only for none DH-enabled ciphers

```
[root@bigip1:ModuleNotLicensed:Active:Standalone] config # ssldump -r /ssl.cap
New TCP connection #1: 10.255.0.80(48710) <-> 10.255.0.202(443)
1 1  0.0009 (0.0009)  C>S  Handshake
     ClientHello
       Version 3.2
       cipher suites
       TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA
       TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA
       Unknown value 0xc022
       Unknown value 0xc021
       TLS_DHE_RSA_WITH_AES_256_CBC_SHA
       TLS_DHE_DSS_WITH_AES_256_CBC_SHA
       TLS_DHE_RSA_WITH_CAMELLIA_256_CBC_SHA
       TLS_DHE_DSS_WITH_CAMELLIA_256_CBC_SHA
       TLS_ECDH_RSA_WITH_AES_256_CBC_SHA
       TLS_ECDH_ECDSA_WITH_AES_256_CBC_SHA
       TLS_RSA_WITH_AES_256_CBC_SHA
       TLS_RSA_WITH_CAMELLIA_256_CBC_SHA
       TLS_ECDHE_RSA_WITH_3DES_EDE_CBC_SHA
       TLS_ECDHE_ECDSA_WITH_3DES_EDE_CBC_SHA
       Unknown value 0xc01c
       Unknown value 0xc01b
       TLS_DHE_RSA_WITH_3DES_EDE_CBC_SHA
       TLS_DHE_DSS_WITH_3DES_EDE_CBC_SHA
       TLS_ECDH_RSA_WITH_3DES_EDE_CBC_SHA
       TLS_ECDH_ECDSA_WITH_3DES_EDE_CBC_SHA
       TLS_RSA_WITH_3DES_EDE_CBC_SHA
       TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA
       TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA
1 2  0.0025 (0.0015)  S>C  Handshake
     ServerHello
       Version 3.2
       session_id[32]=
         da 70 1c cb ae 35 11 2e c2 38 6c d3 16 7d b8 b6
         43 a2 39 c3 9a 66 35 27 0e 94 74 b5 e6 0d 08 c7
       cipherSuite         TLS_DHE_RSA_WITH_AES_256_CBC_SHA
       compressionMethod                   NULL
```

```
1 3  0.0025 (0.0000)  S>C  Handshake
     Certificate
1 4  0.0025 (0.0000)  S>C  Handshake
     ServerKeyExchange
1 5  0.0025 (0.0000)  S>C  Handshake
     ServerHelloDone
1 6  0.0084 (0.0059)  C>S  Handshake
     ClientKeyExchange
       DiffieHellmanClientPublicValue[128]=
         9e 4c ac e6 ab 16 38 7d 00 7d e7 0f ec 22 3c 48
         28 2c 21 d5 67 3e 33 0e 73 bf b5 03 84 ac ad d4
         b0 80 3e 32 90 d3 37 cd 9a 0a 79 17 11 3a 50 20
         39 8b 4f 8b 6d 14 60 84 4f cb b6 a5 be aa 7a ab
         0f a9 49 24 dc 9d 24 bf ee c5 7c f7 3e c8 96 ae
         e2 74 d1 27 83 7a cd b1 2c 74 59 32 6e 00 1c 28
         2d 4c 1d b7 f1 7b 56 44 28 b9 d1 b3 b0 9f d0 88
         3d 17 b0 eb 2f e4 5f 47 58 f5 ac 36 bd e7 e3 5d
```

```
1 24 3.0484 (0.0025)  S>C  application_data
1 25 3.0514 (0.0030)  S>C  application_data
1 26 3.0534 (0.0020)  S>C  application_data
1 27 3.0544 (0.0009)  S>C  application_data
1 28 3.0549 (0.0004)  S>C  application_data
1    3.0549 (0.0000)  S>C  TCP FIN
1 29 3.0554 (0.0005)  C>S  Alert
1    3.0559 (0.0004)  C>S  TCP FIN
```

Show the cipherlist which your cipherstring offers (CLI)



tmm --clientciphers 'default'

```
[root@bigip1:ModuleNotLicensed:Active:Standalone] config # tmm --clientciphers 'default'
      ID  SUITE                          BITS PROT    METHOD  CIPHER    MAC      KEYX
 0:  159  DHE-RSA-AES256-GCM-SHA384      256  TLS1.2  Native  AES-GCM   SHA384   EDH/RSA
 1:  158  DHE-RSA-AES128-GCM-SHA256      128  TLS1.2  Native  AES-GCM   SHA256   EDH/RSA
 2:  107  DHE-RSA-AES256-SHA256          256  TLS1.2  Native  AES       SHA256   EDH/RSA
 3:   57  DHE-RSA-AES256-SHA             256  TLS1    Native  AES       SHA      EDH/RSA
 4:   57  DHE-RSA-AES256-SHA             256  TLS1.1  Native  AES       SHA      EDH/RSA
 5:   57  DHE-RSA-AES256-SHA             256  TLS1.2  Native  AES       SHA      EDH/RSA
 6:   57  DHE-RSA-AES256-SHA             256  DTLS1   Native  AES       SHA      EDH/RSA
 7:  103  DHE-RSA-AES128-SHA256          128  TLS1.2  Native  AES       SHA256   EDH/RSA
 8:   51  DHE-RSA-AES128-SHA             128  TLS1    Native  AES       SHA      EDH/RSA
 9:   51  DHE-RSA-AES128-SHA             128  TLS1.1  Native  AES       SHA      EDH/RSA
10:   51  DHE-RSA-AES128-SHA             128  TLS1.2  Native  AES       SHA      EDH/RSA
11:   51  DHE-RSA-AES128-SHA             128  DTLS1   Native  AES       SHA      EDH/RSA
12:   22  DHE-RSA-DES-CBC3-SHA           168  TLS1    Native  DES       SHA      EDH/RSA
13:   22  DHE-RSA-DES-CBC3-SHA           168  TLS1.1  Native  DES       SHA      EDH/RSA
14:   22  DHE-RSA-DES-CBC3-SHA           168  TLS1.2  Native  DES       SHA      EDH/RSA
15:   22  DHE-RSA-DES-CBC3-SHA           168  DTLS1   Native  DES       SHA      EDH/RSA
16:  157  AES256-GCM-SHA384              256  TLS1.2  Native  AES-GCM   SHA384   RSA
17:  156  AES128-GCM-SHA256              128  TLS1.2  Native  AES-GCM   SHA256   RSA
18:   61  AES256-SHA256                  256  TLS1.2  Native  AES       SHA256   RSA
19:   53  AES256-SHA                     256  TLS1    Native  AES       SHA      RSA
20:   53  AES256-SHA                     256  TLS1.1  Native  AES       SHA      RSA
21:   53  AES256-SHA                     256  TLS1.2  Native  AES       SHA      RSA
22:   53  AES256-SHA                     256  DTLS1   Native  AES       SHA      RSA
23:   60  AES128-SHA256                  128  TLS1.2  Native  AES       SHA256   RSA
24:   47  AES128-SHA                     128  TLS1    Native  AES       SHA      RSA
25:   47  AES128-SHA                     128  TLS1.1  Native  AES       SHA      RSA
26:   47  AES128-SHA                     128  TLS1.2  Native  AES       SHA      RSA
27:   47  AES128-SHA                     128  DTLS1   Native  AES       SHA      RSA
28:   10  DES-CBC3-SHA                   168  TLS1    Native  DES       SHA      RSA
29:   10  DES-CBC3-SHA                   168  TLS1.1  Native  DES       SHA      RSA
30:   10  DES-CBC3-SHA                   168  TLS1.2  Native  DES       SHA      RSA
31:   10  DES-CBC3-SHA                   168  DTLS1   Native  DES       SHA      RSA
```

- Recommended practice for cipherstrings: Don‘t use the default string because the cipherlist depends on the TMOS version. Use a custom cipherstring.

Key Management

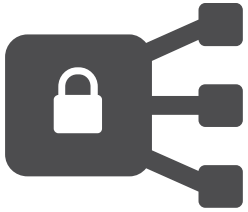
Comprehensive SSL Lifecycle Management

Secure Vault

Software based encrypted storage system for securing cryptographic keys with the highest performance

Internal HSM

Physical hardware designed to generate, store, and protect keys with high performance

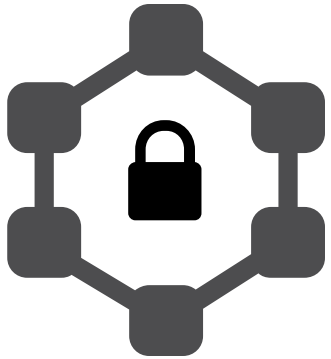


Network HSM

Integration with leading network based hardware for use with all appliances, chassis, and Virtual Editions

Cloud HSM

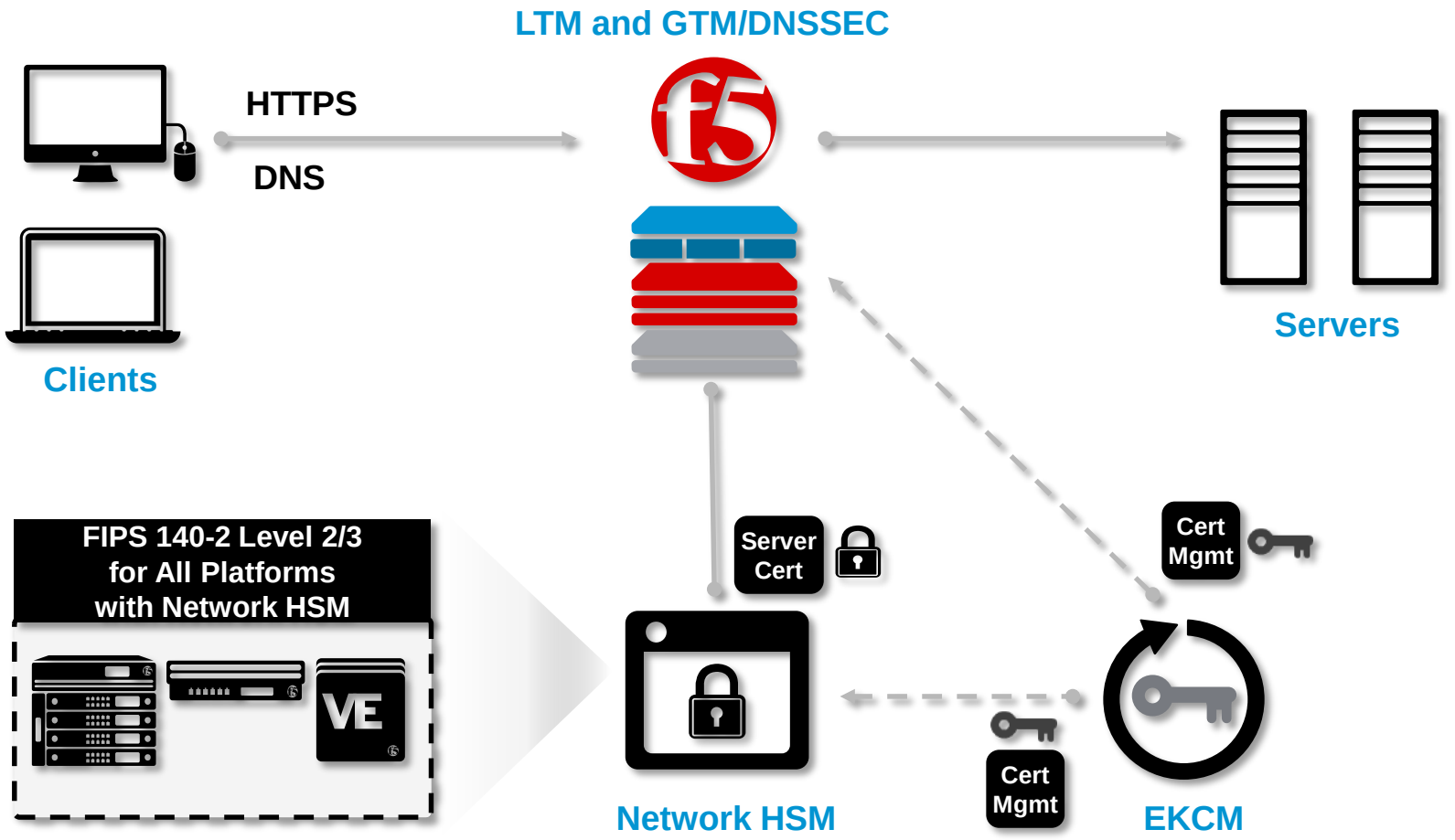
Integration for high-assurance encryption services fit for the cloud.



Enterprise Key and Certificate Management

Open APIs to automate management for the digital certificate and encryption key technologies used by today's enterprises

Hardware Security Module (HSM)



PARTNERSHIPS

Network HSM FIPS140-2 VIPRION, Virtual Edition, and all BIG-IP appliances with Thales nShield Connect HSM, and SafeNet Luna HSM

THALES SafeNet

Automated Enterprise Key and Certificate Management (EK/ EKCM) Venafi and Symantec automate key and certificate management via iControl

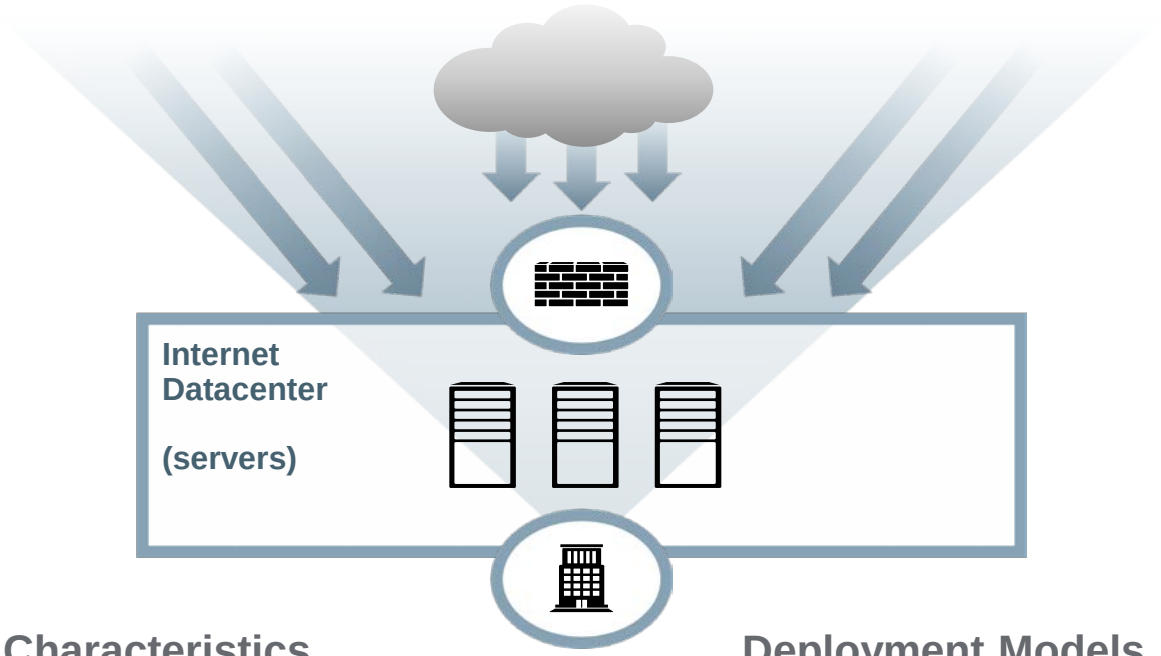
symantec. VENAFI

- Protect keys from eavesdropping or modification
- Supports traditional, virtualized and cloud deployments
- Key management minimizes operational costs and risk

Two main use cases for SSL/TLS

Reverse-Proxy SSL & Forward Proxy SSL

Inbound



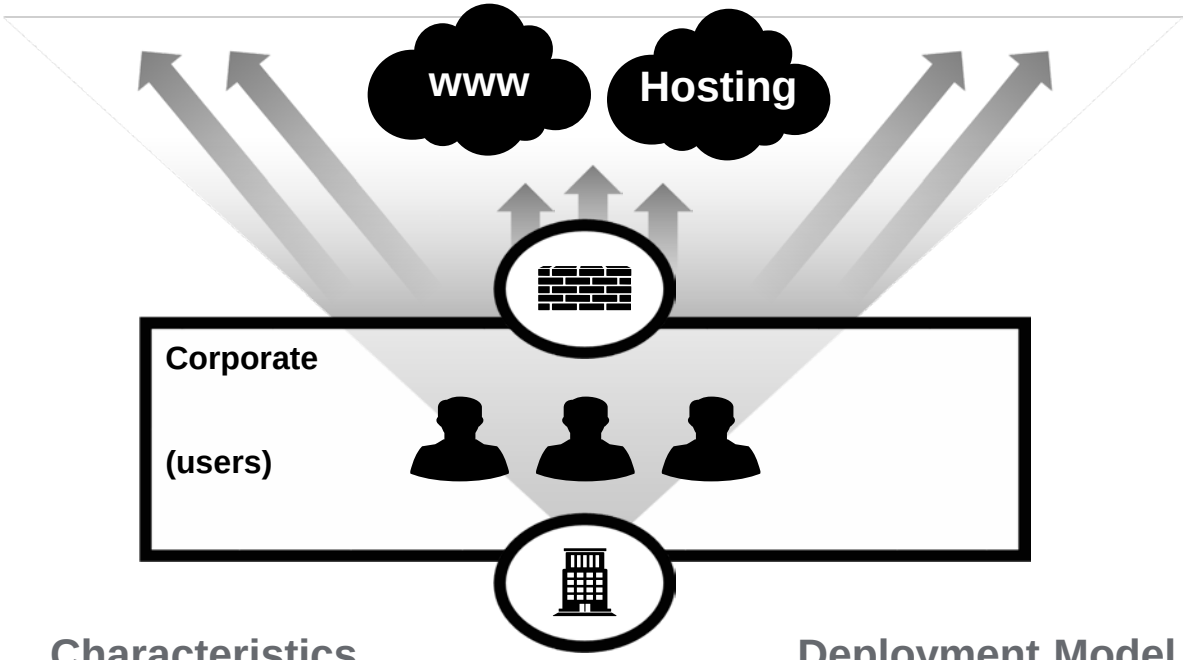
Characteristics

- **Inbound**
- SSL Offload and Acceleration
- Provide visibility for traffic management
- Internet-facing
- Front-end to control and protect access to a server

Deployment Models

- SSL Offload
- SSL Transformation
- Proxy SSL (Split)

Outbound



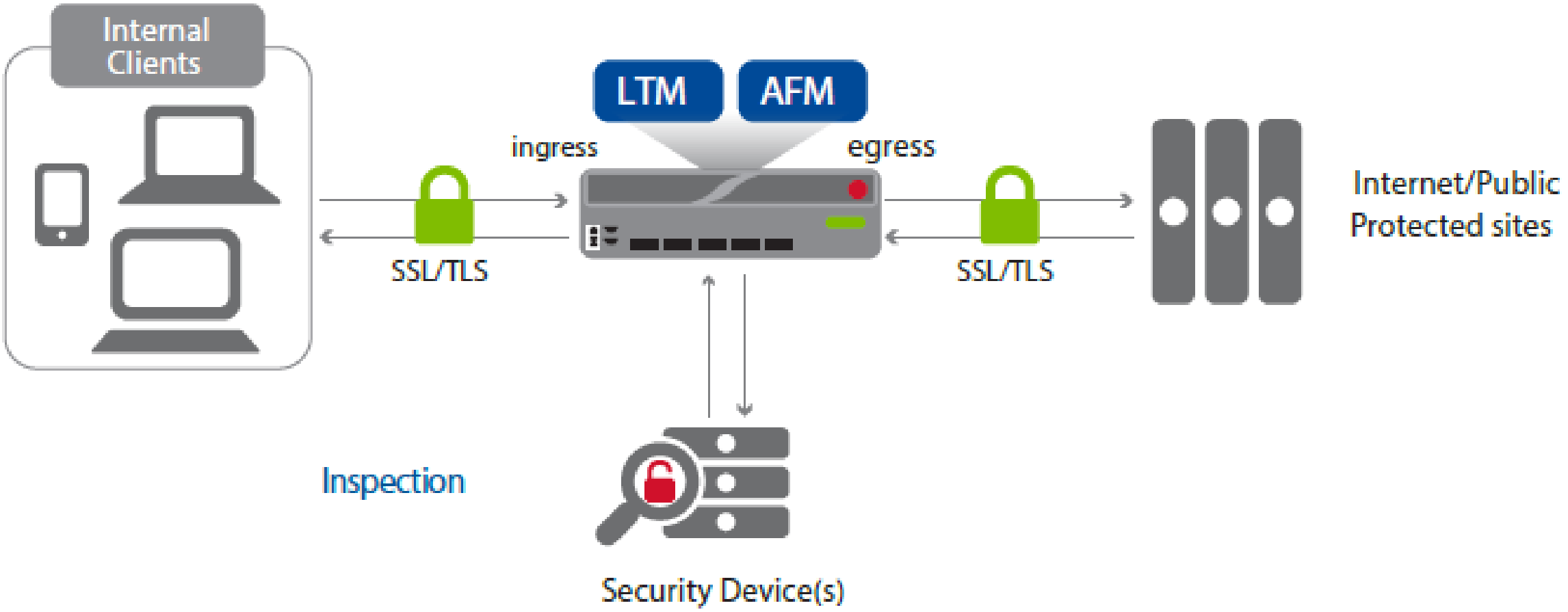
Characteristics

- **Outbound**
- Control user activity
- Sanitize traffic
- Takes requests from an internal network and forwards them to the Internet or Cloud App

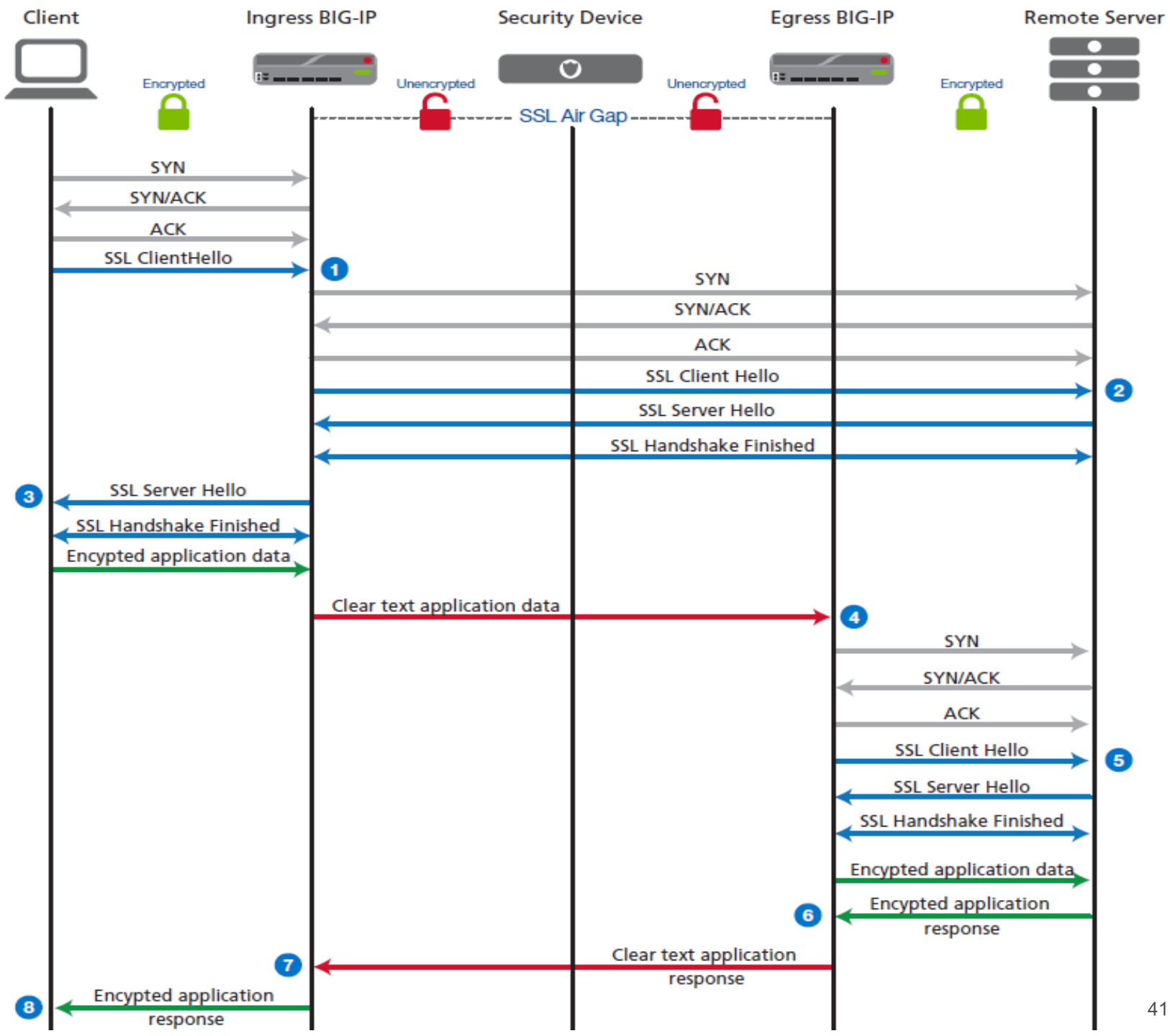
Deployment Model

- SSL Forward Proxy

SSL-/TLS- Intercept – Airgap solution

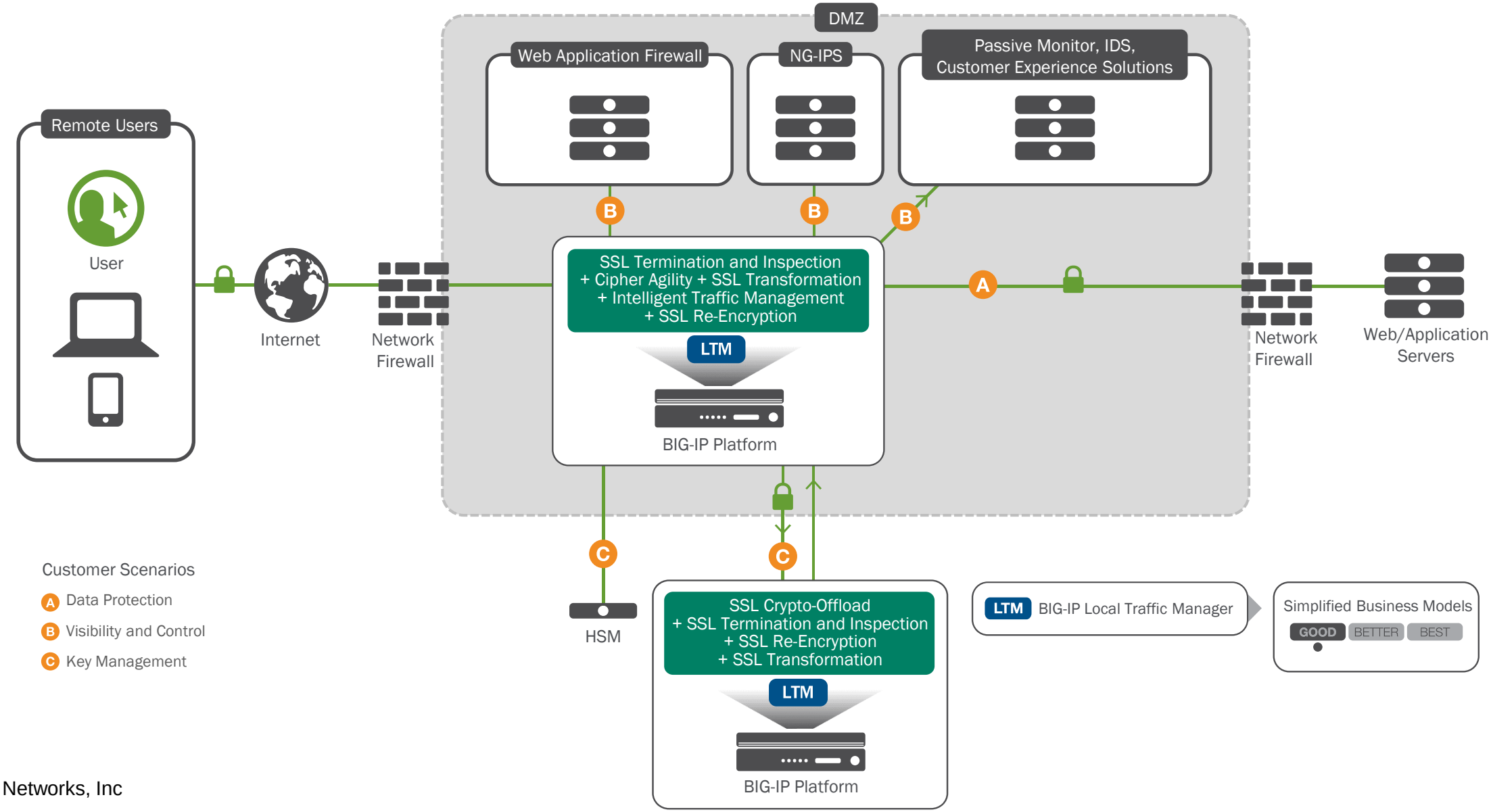


Detailed Traffic-flow for SSL-/TLS-Intercept



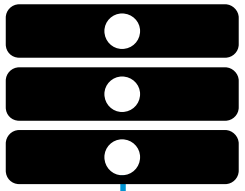
Summary

Encryption without Constraints

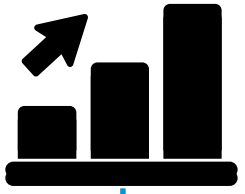


Customer benefit

Maximize investment in security devices



Scale SSL across hybrid environments



Enforce SSL policy on both in *and* outbound traffic

Protect Keys for regulatory compliance

Lowest TCO and quickest ROI

ALL BACKED BY WORLD-CLASS SUPPORT AND PROFESSIONAL SERVICES

SOLUTIONS FOR AN APPLICATION WORLD

SSL